JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 7 (2018) 195-199
Wysłane: 2017-12-09
Przyjęte: 2017-12-19

Porównanie możliwości tworzenia aplikacji internetowych w środowisku JEE na przykładzie Spring Boot i Vaadin

Beniamin Abramowicz*, Beata Pańczyk
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przeanalizowano możliwości tworzenia aplikacji internetowych na platformie Java Enterprise Edition przy zastosowaniu Vaadin i Spring Boot. Do przeprowadzenia analizy opracowano dwie aplikacje testowe typu CRUD, zaimplementowane z wykorzystaniem obu technologii. Przetestowano elementy implementacji, wydajność pracy z bazą danych oraz efektywność ładowania stron w przeglądarce.

Słowa kluczowe: Vaadin; Spring Boot; JEE; aplikacje internetowe

*Autor do korespondencji.

Adres e-mail: beniamin.abramowicz@pollub.edu.pl

Comparison of web applications development possibilities in JEE environment by the example of Spring Boot and Vaadin

Beniamin Abramowicz*, Beata Pańczyk
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The paper presents the analysis of capabilities of creating web applications on the Java Enterprise Edition platform using Spring Boot and Vaadin. To perform the analysis, there were used test applications, implemented in both technologies. Elements of implementation, work efficiency with the database and the efficiency of page loading in the browser were tested.

Keywords: Vaadin; Spring Boot; JEE; web applications

*Corresponding author.

E-mail address: beniamin.abramowicz@pollub.edu.pl

1. Wstęp

Do tworzenia aplikacji internetowych w języku Java wykorzystywana jest platforma Java Enterprise Edition (JEE) oraz najbardziej popularny szkieletem programistyczny Spring MVC [1]. W ostatnich latach pojawiły się nowe rozwiązania mające na celu usprawnienie procesu tworzenia aplikacji na tej platformie. Przykładem takich technologii jest Spring Boot oraz Vaadin.

Vaadin [2] upraszcza tworzenie interfejsu graficznego użytkownika, Spring Boot [3] natomiast automatyzuje proces konfiguracji aplikacji. Obie technologie znajdują się w pierwszej dziesiątce pod względem popularności (Rys. 1) [4].

Vaadin jest frameworkiem zaprojektowanym tak, aby tworzenie i utrzymywanie wysokiej jakości webowych interfejsów użytkownika było łatwe. Obsługuje dwa różne modele programowania: stronę serwera (ang. back end) i stronę klienta (ang. front end). Model serwera pozwala zapomnieć o stronie webowej i programowaniu interfejsu użytkownika [2, 5]. Strona klienta jest wykonywana jako JavaScript w przeglądarce. Nie potrzeba do tego żadnych dodatkowych wtyczek. Vaadin opiera się na wsparciu Google Web Toolkit dla szerokiej rangi przeglądarek, więc programista nie musi się martwić o wsparcie z ich strony.

Spring Boot jest platformą programistyczną stworzoną w celu uproszczenia pisania aplikacji internetowych w Spring MVC. Dzięki niemu konfiguracja aplikacji w celu jej poprawnego uruchomienia i działania zajmuje dużo mniej czasu. Wiele niezbędnych elementów koniecznych do prawidłowej konfiguracji jest uproszczonych do zaledwie kilku linijek kodu.

Java Web Frameworks Index | REBELLABS

Rank	Framework	Popularity
1	Spring MVC	32.00
2	JSF	22.30
3	GWT	9.84
4	Spring Boot	9.45
5	Grails	8.50
6	Struts	7.21
7	Play framework	6.00
8	Vaadin	2.75
9	Dropwizard	1.21
10	JHipster	0.74

Rys 1. Popularność frameworków JEE [4]

2. Cel badań

Do badań wykorzystano dwie aplikacje testowe o tych samych funkcjonalnościach typowej aplikacji CRUD. Aplikacja Vaadin i aplikacja Spring Boot korzystają ze wspólnej bazy danych MySQL. *Back end* obu aplikacji napisany jest w Spring Boot. Vaadin zapewnia *front end*

pierwszej aplikacji testowej. Druga aplikacja wykorzystuje widoki html generowane za pomocą silnika Thymeleaf [6]. Dodatkowo stosuje bibliotekę jQuery oraz Ajax. Do pracy z danymi obie aplikacje korzystają z Hibernate [7]. Pomimo pewnych elementów wspólnych obu aplikacji testowych – implementacja strony serwerowej jest różna.

Celem badań było przeanalizowanie możliwości nowych rozwiązań oraz zbadanie ich wydajności w odniesieniu do aplikacji webowych w środowisku JEE.

3. Aplikacje testowe

Do przygotowania aplikacji testowych wykorzystano:

- Eclipse Oxygen w wersji 4.7.0 – środowisko programistyczne;
- serwer bazy danych MySQLw wersji 5.1.44;
- serwer Apache Tomcat w wersji 8.0;
- Java Enterprise Edition w wersji 8;
- frameworki Vaadin i Spring Boot;
- Hibernate – ORM do pracy z danymi.

Utworzone aplikacje testowe obsługują magazyn hurtowni komputerowej. Umożliwiają:

- wyświetlanie wszystkich produktów z hurtowni;
- dodawanie produktów;
- edycję istniejących produktów;
- usuwanie produktów;
- wyszukiwanie produktów korzystając z różnych kryteriów wyszukiwania.

Oprogramowanie i parametry urządzenia testowego przedstawiono w tabeli 1.

Tabela 1. Parametry i oprogramowanie urządzenia testowego

Parametry/Oprogramowanie	Wersja
System operacyjny	Windows 10 64-bit
Procesor	Intel core i3-2370M 2 rdzenie 2,4 GHz
Pamięć RAM	4 GB DDR3 SDRAM 665MHz
MySQL	5.1.44
Apache Tomcat	8.0
Java	JEE 8

4. Analiza możliwości Vaadin i Spring Boot

W ramach badań:

- porównano struktury obu aplikacji;
- przeprowadzono testy wydajności pracy na bazie danych (pobierania, wyświetlania, zapisywania i wyszukiwania danych w bazie);
- przeprowadzono testy wydajnościowe interfejsu w przeglądarce internetowej;
- porównano podstawowe metryki kodu.

4.1. Struktura aplikacji

Każda z aplikacji posiada charakterystyczną dla siebie strukturę i hierarchię plików. W przypadku technologii Vaadin i Spring Boot występuje dość dużo podobieństw wynikających z modelu aplikacji (Rys. 2):

- **Model aplikacji** – pliki oznaczone kolorem żółtym;
- **Kontroler aplikacji** – pliki oznaczone kolorem jasno zielonym. W przypadku Vaadin kontroler jest zastąpiony prezenterem, którego elementy są zintegrowane z widokami;
- **Widoki aplikacji** – pliki oznaczone kolorem jasno niebieskim;
- **Repozytorium** – pliki oznaczone kolorem czerwonym;
- **Klasa główna uruchamiająca aplikację** – plik oznaczony kolorem jasno różowym;
- **Nawiązanie sesji przy pomocy Hibernate** – pliki oznaczone kolorem ciemno szarym;
- **Style aplikacji** – folder oznaczony kolorem fioletowym;
- **Folder plików wykonywalnych** – folder oznaczony kolorem jasno szarym;
- **Konfiguracja bazy danych i Hibernate** – pliki oznaczone kolorem pomarańczowym;
- **Konfiguracja projektu aplikacji** – plik oznaczony kolorem błękitno-zielonym.

4.2. Wydajność pracy z danymi

Testy wydajności obu aplikacji polegały na pomiarach czasu wykonywania operacji na danych w bazie zgodnie z 5 scenariuszami:

S1 - wyświetlenie wszystkich produktów (100 pomiarów).
S2 - dodanie 100 produktów (100 pomiarów).
S3 - wyszukiwanie po nazwie produktu (10 pomiarów).
S4 - wyszukiwanie po nazwie i typie produktu (10 pomiarów).
S5 - wyszukiwanie po nazwie, typie i zakresie ceny produktu (10 pomiarów).

Przykład 1 pokazuje kod dodający 100 produktów i pomiar czasu dla Vaadin i Spring Boot. Przykład 2 przedstawia kod z pomiarem czasu wyszukania produktu dla Vaadin a przykład 3 – dla Spring Boot.

Przykład 1. Dodanie produktów i pomiar czasu – Vaadin i Spring Boot

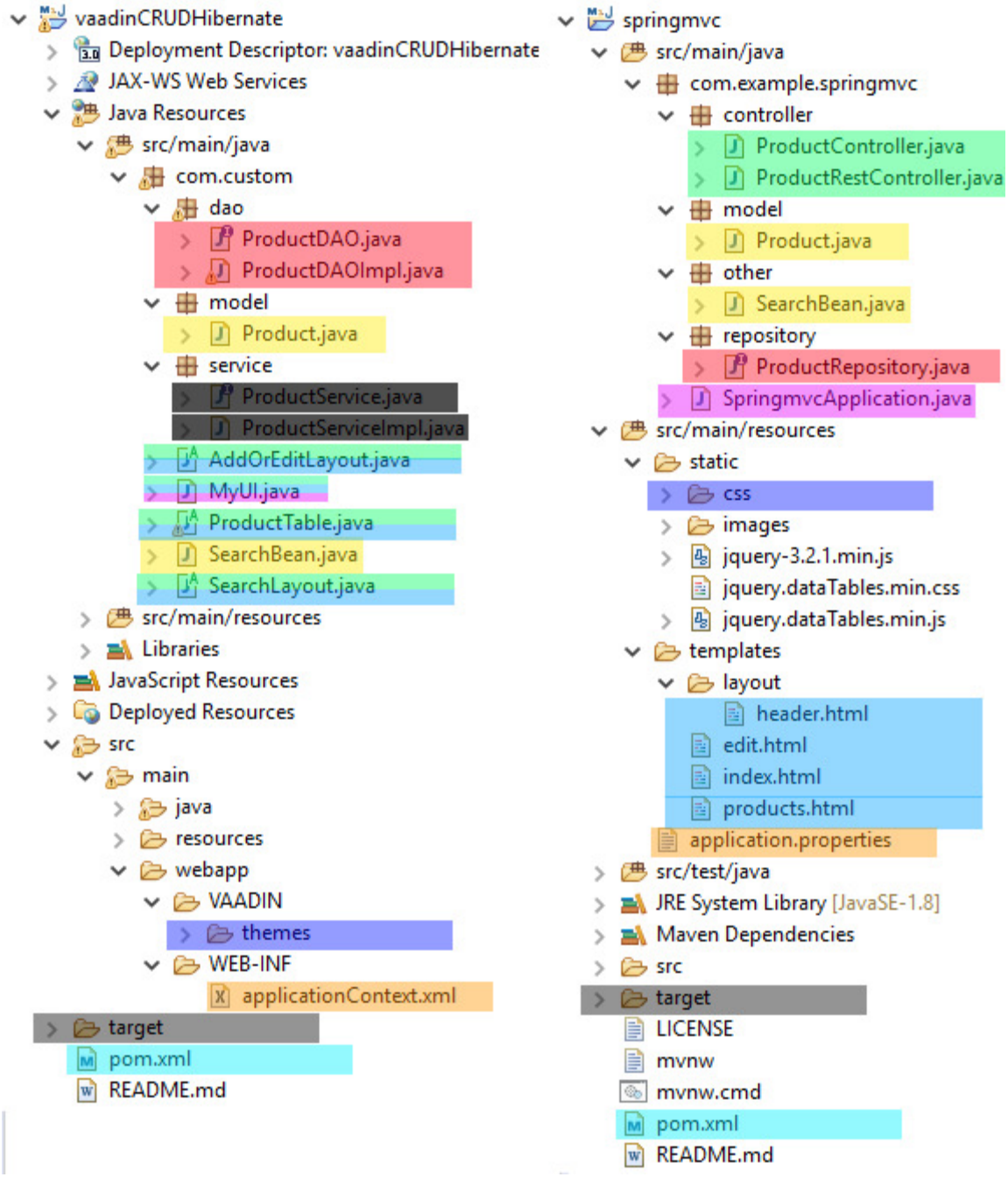
```
long t = System.currentTimeMillis();
for(int i = 0; i<100; i++) {
    Product p = new Product();
    p.setCategory("category");
    p.setDescription("description");
    p.setName("name"+ i);
    p.setPrice(123.0 + i);p.setType("type");
    productService.addProduct(p);
}
long t1 = System.currentTimeMillis();
System.out.println("/product/add100 ->" + (t1-t));
```

Przykład 2. Pomiar czasu wyszukiwania produktu – Vaadin

```
long t = System.currentTimeMillis();
List<Product> a = this.productDao.search(searchBean);
long t2 = System.currentTimeMillis();
System.out.println("/product/search1 ->" + (t2-t));
```

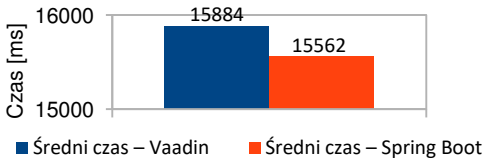
Przykład 3. Pomiar czasu wyszukiwania produktu – Spring Boot

```
Specification<Product>productSpecification=ProductSpecificat
ions.search(searchBean);
List<Product> list =
productRepository.findAll(productSpecification);
long t1 = System.currentTimeMillis();
System.out.println("/product/search1 ->" + (t1-t));
```

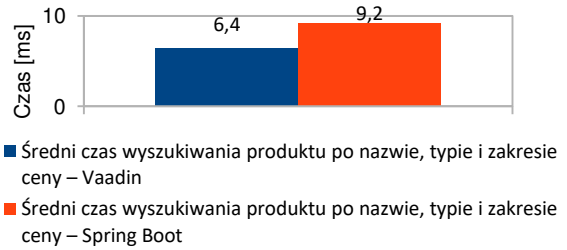


Rys 2. Struktura i hierarchia plików aplikacji Vaadin i Spring Boot

Rysunek 3 przedstawia wykres średniego czasu dodania 100 produktów dla obu aplikacji, natomiast rysunek 4 wykres średniego czasu wyszukiwania po kryteriach: nazwa, typ, zakres ceny.



Rys 3. Średni czas dodania 100 produktów



Rys 4. Średni czas wyszukiwania produktów

W tabeli 2 przedstawiono średni czas dla poszczególnych scenariuszy.

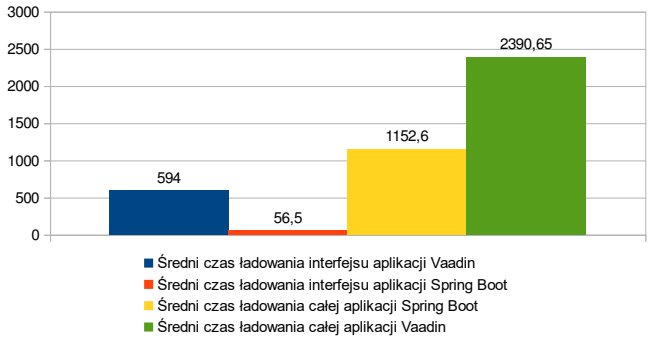
Tabela 2. Średnie czasy operacji na danych dla poszczególnych scrnariuszy

Scenariusz nr	Średnia (Vaadin) [ms]	Średnia (Spring Boot) [ms]
1	13	10
2	15884	15562
3	3,8	6
4	3,5	6
5	6,4	9,2

Analizując wyniki otrzymane z testów można stwierdzić, że wyświetlanie i dodawanie rekordów z bazy jest nieznacznie szybsze w Spring Boot, jednak Vaadin radzi sobie lepiej w wyszukiwaniu produktów nawet dwa razy szybciej.

4.3. Wydajność interfejsu graficznego w przeglądarce

Do wykonania pomiarów wykorzystano ***Performance Panel***, narzędzie deweloperskie dostępne w Google Chrome, które: zbiera dane o poszczególnych modułach ładowanych podczas wejścia na stronę. Na rysunku 5 przedstawiono wykres średnich czasów renderowania interfejsu oraz pobrania wszystkich modułów aplikacji. Wyniki jednoznacznie pokazują, że ładowanie interfejsu graficznego dla Spring Boot jest nawet kilkunastokrotnie szybsze niż dla Vaadin. Wynika to przede wszystkim z tego, że widoki Vaadin są implementowane w języku Java i dopiero podczas uruchamiania aplikacji zostają przetłumaczone na język html. Natomiast widoki Spring Boot, pisane w ThymeLeaf, nie wymagają takiej konwersji.



Rys 5. Średnie czasy renderowania interfejsu i ładowania całej strony dla obu aplikacji

4.4. Metryki kodu

Do zliczenia linii kodu obu aplikacji wykorzystano program LocMetrics [8]. Tworzy on logi, w których liczba linii kodu jest wyświetlona z podziałem na poszczególne pliki, pomijając puste linie i komentarze. W tabeli 3 przedstawiono liczbę linii kodu dla każdej z aplikacji.

Tabela 3. Metryki kodu z podziałem na pliki dla Vaadin i Spring Boot

Nazwa pliku	Rola pliku	Liczba linii kodu Vaadin	Liczba linii koduSpring Boot
Product.Java	Model	71	74
SearchBean.java	Model	86	98
ProductController.java	Kontroler	-	59
ProductRestController.java	Kontroler	-	188
SpringmvcApplication.java	Klasa główna uruchamiająca aplikację	-	13
MyUI.java	Klasa główna uruchamiająca aplikację/widok	175	-
ProductTable.java	Widok	108	-
AddOrEditLayout.java	Widok	71	-
SearchLayout.java	Widok	112	-
Header.html	Widok	-	39
Edit.html	Widok	-	56
Index.html	Widok	-	10
Products.html	Widok	-	211
ProductDAO.java	Interfejs	20	-
ProductDAOImpl.java	Repozytorium/implementacja interfejsu	136	-
ProductRepository.java	Repozytorium	-	14
ProductService.java	Interfejs	24	-
ProductServiceImpl.java	Operacje na bazie/implementacja interfejsu	88	-
ApplicationContext.xml	Konfiguracja bazy danych i Hibernate	39	-
Applications.properties	Konfiguracja bazy danych i Hibernate	-	7
Pom.xml	Konfiguracja projektu	308	74
SpringmvcApplicationTest.java	Klasa testowa sprawdzająca Spring Boot	-	18
Razem linii kodu	-	**1238**	**861**
Waga pliku wykonywalnego [MB]	-	**43.97**	**35.63**

Aplikacja w Spring Boot zajmuje o około 1/3 mniej kodu niż aplikacja w Vaadin. Jest to spowodowane przede wszystkim uproszczeniem konfiguracji aplikacji do minimum: konfiguracja projektu, połączenia z bazą danych, Hibernate, repozytoriów. W konsekwencji również plik wykonywalny aplikacji Spring Boot ma mniejszy rozmiar o prawie 10 MB.

5. Ocena Spring Boot i Vaadin

Subiektywna ocena autorów obu frameworków przedstawiona jest w tabeli 4. Zastosowano skalę ocen 1-5, gdzie 5 ocena najwyższa..

Tabela 4. Ocena obu technologii

Kryterium	Vaadin	Spring Boot
Dostępność do materiałów/poradników	3	4
Dostęp do narzędzi programistycznych	5	5
Przejrzystość struktury plików aplikacji	4	5
Przejrzystość modelu tworzenia aplikacji	3	4
Łatwość implementacji interfejsu graficznego	5	4
Łatwość implementacji zaplecza aplikacji	2	4
Łatwość konfiguracji	3	5
Wygoda tworzenia stylów aplikacji	5	3
Wydajność wyświetlania rekordów z bazy	3	5
Wydajność dodawania rekordów do bazy	4	4
Wydajność wyszukiwania rekordów używając różnych kryteriów wyszukiwania	5	4
Szybkość renderowania interfejsu	2	5
Szybkość uruchamiania aplikacji	2	5
Liczba linii kodu potrzebna do stworzenia w pełni funkcjonalnej i działającej aplikacji (5 - bardzo mało, 1 - bardzo dużo)	3	4
Ocena końcowa	**49**	**61**

6. Wnioski

Technologie Spring Boot i Vaadin pozwalają na szybkie tworzenie aplikacji internetowych. Spring Boot ułatwia konfigurację aplikacji, połączenie z bazą danych oraz konfigurację repozytorium, podczas gdy Vaadin przyspiesza tworzenie widoków dzięki zastosowaniu języka Java, a także zawarciu prezentera, czyli pośrednika pomiędzy modelem a widokami. Dzięki temu wyszukiwania elementów po określonych kryteriach i wyświetlenie wyników odbywa się szybciej niż w Spring Boot. Jednocześnie tworzenie widoków bezpośrednio w języku Java ma też wady, a mianowicie widoki muszą być konwertowane z języka Java do HTML co wydłuża renderowanie interfejsu i ładowanie całej strony. Spring Boot jest szybszy między innymi dlatego, iż widoki mogą być pisane w HTML za pomocą ThymeLeaf. Każda z technologii ułatwia programowanie aplikacji internetowych w JEE. Wybór zależy w dużej mierze od tego czy potrzebna jest szybka konfiguracja bazy i szybkie wczytywanie stron, czy prosty do implementacji interfejs użytkownika.

Literatura

[1] http://www.oracle.com/technetwork/java/javaee/tech/index.html [11.10.2017]

[2] https://vaadin.com/docs [1.12.2017]

[3] https://spring.io/guides/gs/spring-boot/, [1.12.2017]

[4] https://zeroturnaround.com/webframeworksindex/ [14.12.2017]

[5] https://demo.vaadin.com/book-examples/book, 12.03.2017

[6] http://www.thymeleaf.org/doc/tutorials/2.1/thymeleafspring.html, [16.12.2017]

[7] http://hibernate.org/, [16.12.2017]

[8] http://www.locmetrics.com [14.12.2017]